UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/822,938	03/30/2001	Ronny Ronen	2207/8610	8312

26646 7590 10/12/2004

KENYON & KENYON
ONE BROADWAY
NEW YORK, NY 10004

EXAMINER	
GERSTL, SHANE F	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 10/12/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/822,938	RONEN ET AL.	
	Examiner	Art Unit	
	Shane F Gerstl	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 July 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☒ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-33 have been examined.

Papers Received

2. Receipt is acknowledged of amendment and extension of time papers submitted, where the papers have been placed of record in the file.

3. The amendment has successfully overcome the objections to the drawings and title, both of which are herein withdrawn.

4. The objection to the Oath/Declaration remains as set forth below.

Oath/Declaration

5. The oath or declaration is defective. A new oath or declaration in compliance with 37 CFR 1.67(a) identifying this application by application number and filing date is required. See MPEP §§ 602.01 and 602.02.

> The oath or declaration is defective because:
> It does not identify the mailing or post office address of each inventor, but only the residence. If it is the intention of the declaration to have the post office address be the same as the residence, then it must be indicated so in some form. A mailing address is an address at which an inventor customarily receives his or her mail and may be either a home or business address. The mailing address should include the ZIP Code designation. The mailing address may be provided in an application data sheet or a supplemental oath or declaration. See 37 CFR 1.63(c) and 37 CFR 1.76.
>
> Non-initialed and/or non-dated alterations have been made to the oath or declaration. See 37 CFR 1.52(c). In this case, the residence of Adi Yoaz has been altered without being initialed and dated.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claim 32 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. It is unclear from the claim language whether it is meant for the first and second source information to each include exactly three bits or for the first and second information to include exactly three bits between the two of them. The examiner is taking the claim to mean that each of the source information includes exactly three bits based on the specification.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9. Claims 1-33 are rejected under 35 U.S.C. 102(b) as being anticipated by Vajapeyam et al.

10. In regard to claim 1, Vajapeyam has disclosed a method for renaming a source for use with a processor, the method comprising:

a. providing at least one instruction; [Page 1, column 2 shows that instructions are fetched and decoded and thus provided.]

b. building instruction dependency information based on the at least one instruction; [In the same paragraph as cited directly above, it is shown that these fetched and decoded instructions are renamed. Since renaming is used to

resolve dependencies (as shown in the abstract as the underlying goal) the dependency information is gathered or built to accommodate this renaming.]

c. caching the at least one instruction with the instruction dependency information to provide cached instruction information; [Page 2, column 2 shows that register renaming (dependency) information is recorded in the trace cache. Figure 5 shows that the trace cache entries hold the instruction and the associated dependency or rename information.]

d. renaming a register based on the cached instruction information to provide a renamed register; [Page 5, section 2.3 shows that a lower-bandwidth single cycle rename stage processes information from the cache. One of ordinary skill in the art would recognize that the rename stage inherently produces a renamed register.]

e. And multiplexing the instruction dependency information and the renamed register to rename the source. [Page 4, column 2 shows that a local mapping is forwarded a subsequent map modification stage or a mapping is picked up from a global free list (table). It is shown here that the selection between these two options is made by a multiplexer. The mapping that is picked up from the global free list is the renamed register as described by the process for traditional register renaming at the top of the paragraph. The local mapping is dependency information for the instruction that is sent to another stage for modification before global renaming.]

11. In regard to claim 2, Vajapeyam discloses the method of claim 1, wherein the cached instruction information includes first source information, second source information and destination information (figure 5).

12. In regard to claim 3, Vajapeyam discloses the method of claim 2, wherein the first source information includes three bits, the second source information includes three bits, and the destination information includes one bit. [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

13. In regard to claim 4, Vajapeyam discloses the method of claim 1, wherein the first source information and the second source information denote a rename window instruction from which the at least one instruction depends. [Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it as the claim suggests. Vajapeyam shows that all instructions have two sources as shown in figure 5. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

14. In regard to claim 5, Vajapeyam discloses the method of claim 1, wherein a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6

shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

15. In regard to claim 6, Vajapeyam discloses the method of claim 1, wherein:

a. the first source information includes three bits, the second source information includes three bits, and the destination information includes one bit; [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

b. the first source information and the second source information denote a rename window instruction from which the at least one instruction depends; [Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it. All instructions have two sources as shown in figure 5 of Vajapeyam. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

c. and a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6 shows the renamer interacting with a line from

the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

16. In regard to claim 7, Vajapeyam discloses the method of claim 6, wherein the processor is a microprocessor. [The IEEE dictionary definition included herein defines a "microprocessor" to be "an integrated circuit that contains the logic elements for manipulating data and for making decisions." Thus the superscalar processor taught by Vajapeyam is a microprocessor.]

17. In regard to claim 8, Vajapeyam discloses a system for renaming a source for use with a processor, the system comprising:

a. a fetch and decoding arrangement for fetching and decoding at least one instruction from the processor (figure 2);

b. a build-instruction-dependency arrangement for building instruction dependency information based on the at least one instruction; [On page 1, column 2, it is shown that these fetched and decoded instructions are renamed. Since renaming is used to resolve dependencies (as shown in the abstract as the underlying goal) the dependency information is gathered or built to accommodate this renaming.]

c. an instruction cache arrangement for caching the at least one instruction with the instruction dependency information to provide cached instruction information, the build-instruction-dependency arrangement providing the

instruction dependency information to the instruction cache arrangement; [Page 2, column 2 shows that register renaming (dependency) information is recorded in a trace cache (of figure 2). Figure 5 shows that the trace cache entries hold the instruction and the associated dependency information.]

d. a renamer arrangement for renaming a register based on the cached instruction information and for providing a renamed register; [Figure 2 shows a remap unit (renamer) that renames registers of instructions fetched from the cache.]

e. and a multiplexing arrangement for multiplexing the instruction dependency information and the renamed register and for providing a renamed source. [Page 4, column 2 shows that a local mapping is forwarded a subsequent map modification stage or a mapping is picked up from a global free list (table). It is shown here that the selection between these two options is made by a multiplexer. The mapping that is picked up from the global free list is the renamed register as described by the process for traditional register renaming at the top of the paragraph. The local mapping is dependency information for the instruction that is sent to another stage for modification before global renaming.]

18. In regard to claim 9, Vajapeyam discloses the system of claim 8, wherein the cached instruction information includes first source information, second source information and destination information (figure 5).

19. In regard to claim 10, Vajapeyam discloses the system of claim 9, wherein the first source information includes three bits, the second source information includes three

bits, and the destination information includes one bit. [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

20. In regard to claim 11, Vajapeyam discloses the system of claim 8, wherein the first source information and the second source information denote a rename window instruction from which the at least one instruction depends. [Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it. All instructions have two sources as shown in figure 5 of Vajapeyam. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

21. In regard to claim 12, Vajapeyam discloses the system of claim 8, wherein a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6 shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

22. In regard to claim 13, Vajapeyam discloses the system of claim 8, wherein:

a. the first source information includes three bits, the second source information includes three bits, and the destination information includes one bit; [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

b. the first source information and the second source information denote a rename window instruction from which the at least one instruction depends; [Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it. All instructions have two sources as shown in figure 5 of Vajapeyam. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

c. and a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6 shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

23. In regard to claim 14, Vajapeyam discloses the method of claim 13, wherein the processor is a microprocessor. [The IEEE dictionary definition included herein defines a "microprocessor" to be "an integrated circuit that contains the logic elements for manipulating data and for making decisions." Thus the superscalar processor taught by Vajapeyam is a microprocessor.]

24. In regard to claim 15, Vajapeyam discloses a system for renaming a source for use with a processor, the system comprising:

a. Means for fetching and decoding at least one instruction from the processor (figure 2);

b. Means for building instruction dependency information based on the at least one instruction; [On page 1, column 2, it is shown that these fetched and decoded instructions are renamed. Since renaming is used to resolve dependencies (as shown in the abstract as the underlying goal) the dependency information is gathered or built to accommodate this renaming.]

c. Means for caching the at least one instruction with the instruction dependency information to provide cached instruction information, the build-instruction-dependency arrangement providing the instruction dependency information to the instruction cache arrangement; [Page 2, column 2 shows that register renaming (dependency) information is recorded in a trace cache (of figure 2). Figure 5 shows that the trace cache entries hold the instruction and the associated dependency information.]

d. Means for renaming a register based on the cached instruction information and for providing a renamed register; [Figure 2 shows a remap unit (renamer) that renames registers of instructions fetched from the cache.]

e. And means for multiplexing the instruction dependency information and the renamed register and for providing a renamed source. [Page 4, column 2 shows that a local mapping is forwarded a subsequent map modification stage or a mapping is picked up from a global free list (table). It is shown here that the selection between these two options is made by a multiplexer. The mapping that is picked up from the global free list is the renamed register as described by the process for traditional register renaming at the top of the paragraph. The local mapping is dependency information for the instruction that is sent to another stage for modification before global renaming.]

25. In regard to claim 16, Vajapeyam discloses the system of claim 15, wherein the cached instruction information includes first source information, second source information and destination information (figure 5).

26. In regard to claim 17, Vajapeyam discloses the system of claim 16, wherein the first source information includes three bits, the second source information includes three bits, and the destination information includes one bit. [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

27. In regard to claim 18, Vajapeyam discloses the system of claim 15, wherein the first source information and the second source information denote a rename window instruction from which the at least one instruction depends. [Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it. All instructions have two sources as shown in figure 5 of Vajapeyam. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

28. In regard to claim 19, Vajapeyam discloses the system of claim 15, wherein a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6 shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

29. In regard to claim 20, Vajapeyam discloses the system of claim 15, wherein:

a. the first source information includes three bits, the second source information includes three bits, and the destination information includes one bit; [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

b. the first source information and the second source information denote a rename window instruction from which the at least one instruction depends; [Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it. All instructions have two sources as shown in figure 5 of Vajapeyam. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

c. and a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6 shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

30. In regard to claim 21, Vajapeyam discloses the method of claim 15, wherein the processor is a microprocessor. [The IEEE dictionary definition included herein defines a "microprocessor" to be "an integrated circuit that contains the logic elements for manipulating data and for making decisions." Thus the superscalar processor taught by Vajapeyam is a microprocessor.]

31. [In regard to claim 22, the examiner is taking "a storage medium" to be a computer-readable medium because the set of instructions does not appear to have any other utility in this instance than in a computer and is thus statutory.] Vajapeyam has disclosed a set of instructions residing in a storage medium, said set of instructions capable of being executed by a processor to implement a method for renaming a source for use with a processor, (Vajapeyam's disclosed processor uses instructions stored in a cache as shown below) the method comprising:

a. providing at least one instruction; [Page 1, column 2 shows that instructions are fetched and decoded and thus provided.]

b. building instruction dependency information based on the at least one instruction; [In the same paragraph as cited directly above, it is shown that these fetched and decoded instructions are renamed. Since renaming is used to resolve dependencies (as shown in the abstract as the underlying goal) the dependency information is gathered or built to accommodate this renaming.]

c. caching the at least one instruction with the instruction dependency information to provide cached instruction information; [Page 2, column 2 shows that register renaming (dependency) information is recorded in the trace cache. Figure 5 shows that the trace cache entries hold the instruction and the associated dependency information.]

d. renaming a register based on the cached instruction information to provide a renamed register; [Page 5, section 2.3 shows that a lower-bandwidth single cycle rename stage processes information from the cache. One of ordinary skill

in the art would recognize that the rename stage inherently produces a renamed register.]

e. And multiplexing the instruction dependency information and the renamed register to rename the source. [Page 4, column 2 shows that a local mapping is forwarded a subsequent map modification stage or a mapping is picked up from a global free list (table). It is shown here that the selection between these two options is made by a multiplexer. The mapping that is picked up from the global free list is the renamed register as described by the process for traditional register renaming at the top of the paragraph. The local mapping is dependency information for the instruction that is sent to another stage for modification before global renaming.]

32. In regard to claim 23, Vajapeyam discloses the method of claim 22, wherein the cached instruction information includes first source information, second source information and destination information (figure 5).

33. In regard to claim 24, Vajapeyam discloses the method of claim 23, wherein the first source information includes three bits, the second source information includes three bits, and the destination information includes one bit. [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

34. In regard to claim 25, Vajapeyam discloses the method of claim 22, wherein the first source information and the second source information denote a rename window

instruction from which the at least one instruction depends. [Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it. All instructions have two sources as shown in figure 5 of Vajapeyam. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

35. In regard to claim 26, Vajapeyam discloses the method of claim 22, wherein a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6 shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

36. In regard to claim 27, Vajapeyam discloses the method of claim 22, wherein:

a. the first source information includes three bits, the second source information includes three bits, and the destination information includes one bit; [Figure 5 shows that the first and second source information contain 4 fields and thus, logically, include at least three bits. The figure also shows that the destinations information includes 3 fields and therefore includes at least one bit.]

b. the first source information and the second source information denote a rename window instruction from which the at least one instruction depends;

[Applicant has not defined what a rename window actually is in the specification, but merely uses it. Therefore, the examiner is taking a rename window instruction to be any instruction containing two sources with another instruction depending on it. All instructions have two sources as shown in figure 5 of Vajapeyam. Since these instructions were renamed, another instruction must have a dependency on each one and is thus a rename window instruction (or trace window instructions as shown in figure 2).]

c. and a virtual ID is formed by at least one of adding, combining, concatenating or logically OR-ing upper bits from a renamer with lower bits from an instruction cache. [Figure 6 shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. Thus, essentially, bits (lower bits) from the cache are combined with bits (upper bits) from the renamer.]

37. In regard to claim 28, Vajapeyam discloses the method of claim 27, wherein the processor is a microprocessor. [The IEEE dictionary definition included herein defines a "microprocessor" to be "an integrated circuit that contains the logic elements for manipulating data and for making decisions." Thus the superscalar processor taught by Vajapeyam is a microprocessor.]

38. In regard to claim 29, Vajapeyam discloses the method of claim 1, wherein the instruction dependency information indicates upon which other instruction the at least one instruction depends. [The example of Figure 3 shows that the renamed i2

instruction indicates that the architectural register for source 2 is R1 and the physical register for this source is 2. Since the source is renamed, this indicates that a dependency exists before this instruction in program order and in the figure specifically indicates that this instruction would be instruction i1.]

39. In regard to claim 30, Vajapeyam discloses the method of claim 1, wherein the renamed register is a virtual renamed register. [Figure 6 shows the renamer interacting with a line from the trace cache (instruction cache). The trace cache sends data to a MAP TBL (part of the renamer) where based on the data it receives, a renamed register or virtual ID is formed. The renamed register or virtual ID refers to a virtual register.]

40. In regard to claim 31, Vajapeyam discloses the method of claim 30, wherein the virtual renamed register refers to a value that is to be produced in the future by the at least one instruction. [When renamed register is mapped to the destination register, the value referred to in the register has yet to execute and will occupy once execution completes.]

41. In regard to claim 32, Vajapeyam discloses the method of claim 2, wherein the first and second source information includes exactly three bits. [The claim uses open-ended and non-limiting language with "wherein" and "includes". Specifically, the term "includes" means that the details following the term are included but does not limit other details from being included. Even with the adverb "exactly" in the claim language, the first and second source information need include exactly three bits, but can also include more. As shown in figures 3 and 5, the first and second source information consists of 4 fields and therefore comprises at least 4 bits. Thus the first and second source

information includes exactly three bits and also includes at least one more bit. If it is meant by the applicant to claim that the source information consists of exactly three bits (closed-form and limiting to only three), the Examiner recommends using "consisting of" language rather than "including" language.]

42. In regard to claim 33, Vajapeyam discloses the method of claim 2, wherein the destination information includes exactly one bit. [The claim uses open-ended and non-limiting language with "wherein" and "includes". Specifically, the term "includes" means that the details following the term are included but does not limit other details from being included. Even with the adverb "exactly" in the claim language, the destination information need include exactly one bit, but can also include more. As shown in figures 3 and 5, the destination information consists of 3 fields and therefore comprises at least 3 bits. Thus the first and second source information includes exactly one bit and also includes at least two more bits. If it is meant by the applicant to claim that the destination information consists of exactly one bit (closed-form and limiting to only one), the Examiner recommends using "consisting of" language rather than "including" language.]

Response to Arguments

43. Applicant's arguments filed 7/1/04 have been fully considered but they are not persuasive.

44. Applicant has argued that the Vajapayem reference records register renaming information in a trace cache and that this is in contrast to claim 1, which records dependency information in the trace cache. The examiner agrees that Vajapayem

records register renaming information the trace, but maintains that this renaming information is dependency information. As shown in figure 5 and in previously cited sections such as section 2.3, the trace cache stores renamed register identifiers. One of ordinary skill in the art would have recognized that register renaming, by definition, is done so as to avoid hazards, which are the results of dependencies and that this technique is old and known in the art. Thus the renamed registers avoid the hazards by solving the dependencies, and thus this register renaming information is in fact dependency information. The examiner has included recitations from a textbook (Computer Architecture: A Quantitative Approach by Hennessy and Patterson; hereafter referred to as Hennessy) further illustrating this definition. Page 232 of Hennessy shows that by definition register renaming solves naming dependencies and hazards (illustrated on page 233) and thus the renamed registers and their identifiers are dependency information. Pages 251-261 give an old system making practical use of the defined register renaming technique. Page 252 specifically shows that register renaming avoids hazards and resolves dependencies by renaming registers and thus the renaming information is dependency information.

45. Applicant has argued that claims 8, 15, and 22 recite essential the same limitations as claim 1 and thus the same arguments given above apply.

46. The dependent claims remain rejected as given above due to their dependency on the independent claims and due to the rejections of their individual limitations given above.

Conclusion

47. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

48. The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

49. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. The references cited in the previous action remain pertinent.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Shane F Gerstl whose telephone number is (571) 272-4166 after October 12 and (703) 305-7305 before October 12. The examiner can normally be reached on M-F 6:45-4:15 (First Friday Off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162 after October 12 and (703) 305-9712 before October 12. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Shane F Gerstl
Examiner
Art Unit 2183

SFG
October 5, 2004

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100